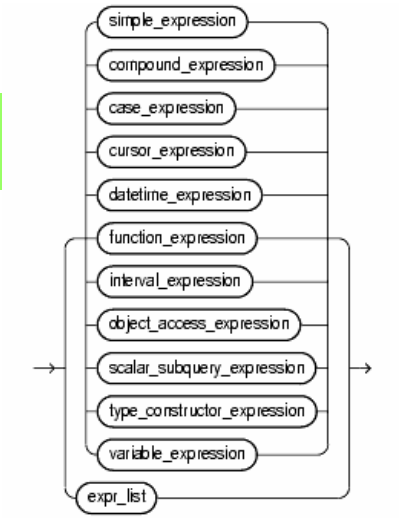
Chapitre D

Expressions et conditions SQL

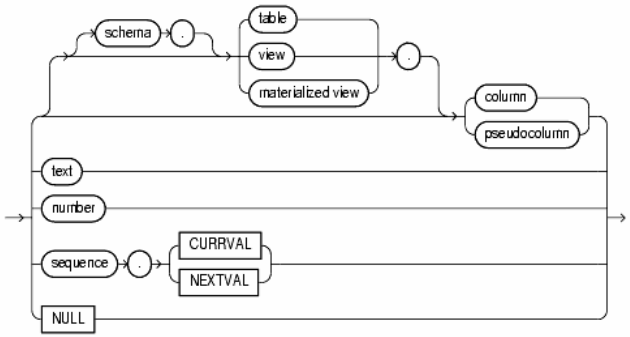
Expression

- Simple expression
  - 7.8/2
  - **`TO_CHAR (TRUNC(SYSDATE+7))`**
- Utilisées
  - clauses **`SELECT`** niveau **`WHERE, HAVING`**
  - **`VALUES`** dans **`INSERT`**
  - **`SET`** dans **`UPDATE`**

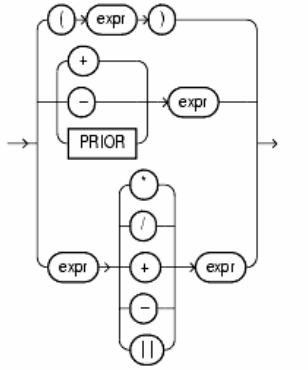
Expression



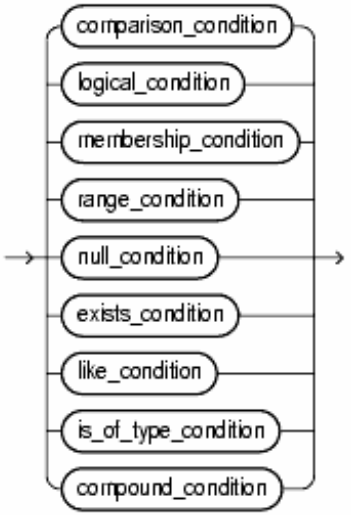
Simple expression



Compound Expression



condition



Priorités

Condition	Purpose
SQL operators	See "Operator Precedence"
=, !=, <, >, <=, >=,	comparison
IS [NOT] NULL, LIKE, [NOT] BETWEEN, [NOT] IN, EXISTS, IS OF *type*	comparison
NOT	exponentiation, logical negation
AND	conjunction
OR	disjunction

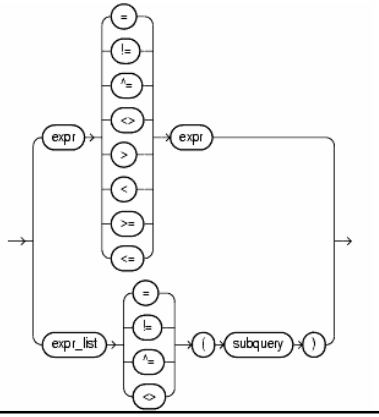
Comparison conditions

- **TRUE, FALSE, UNKNOWN**
- Problèmes des LOBs
  - comparaison impossible en SQL
  - possible en PL/SQL
- Exemples

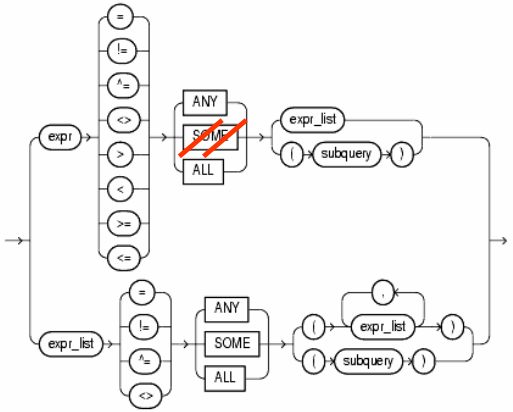
Condition	Purpose	Example
=	Equality test.	SELECT * FROM employees WHERE salary = 2500;
!= ^= <> ¬=	Inequality test. Some forms of the inequality condition may be unavailable on some platforms.	SELECT * FROM employees WHERE salary != 2500;
>	"Greater than" and "less than" tests.	SELECT * FROM employees WHERE salary > 2500;
<		SELECT * FROM employees WHERE salary < 2500;
>=	"Greater than or equal to" and "less than or equal to" tests.	SELECT * FROM employees WHERE salary >= 2500;
<=		SELECT * FROM employees WHERE salary <= 2500;

ANY SOME	Compares a value to each value in a list or returned by a query. Must be preceded by =, !=, >, <, <=, >=. Evaluates to FALSE if the query returns no rows.	SELECT * FROM employees WHERE salary = ANY (SELECT salary FROM employees WHERE department_id = 30);
ALL	Compares a value to every value in a list or returned by a query. Must be preceded by =, !=, >, <, <=, >=. Evaluates to FALSE if the query returns no rows.	SELECT * FROM employees WHERE salary >= ALL (1400, 3000);

Simple comparison condition



Group comparison condition



Logical condition

Condition	Operation	Example
NOT	Returns TRUE if the following condition is FALSE. Returns FALSE if it is TRUE. If it is UNKNOWN, it remains UNKNOWN.	SELECT * FROM employees WHERE NOT (job_id IS NULL); SELECT * FROM employees WHERE NOT (salary BETWEEN 1000 AND 2000);
AND	Returns TRUE if both component conditions are TRUE. Returns FALSE if either is FALSE. Otherwise returns UNKNOWN.	SELECT * FROM employees WHERE job_id = 'PU_CLERK' AND department_id = 30;
OR	Returns TRUE if either component condition is TRUE. Returns FALSE if both are FALSE. Otherwise returns UNKNOWN.	SELECT * FROM employees WHERE job_id = 'PU_CLERK' OR department_id = 10;

Table 5-4 NOT Truth Table

	TRUE	FALSE	UNKNOWN
NOT	FALSE	TRUE	UNKNOWN

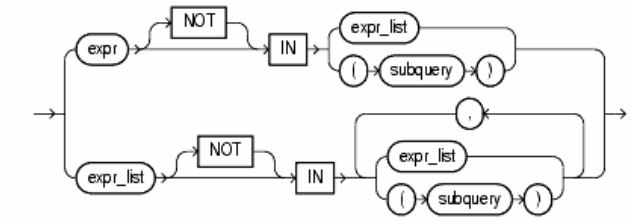
Table 5-5 AND Truth Table

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

Table 5-6 OR Truth Table

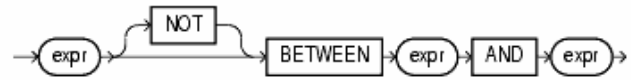
OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

Membership conditions

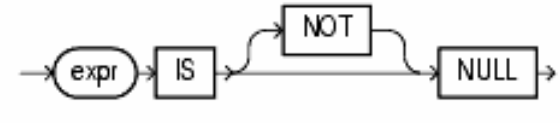


Condition	Operation	Example
IN	"Equal to any member of" test. Equivalent to "= ANY".	SELECT * FROM employees WHERE job_id IN ('PU_CLERK','SH_CLERK'); SELECT * FROM employees WHERE salary IN (SELECT salary FROM employees WHERE department_id =30);
NOT IN	Equivalent to "!=ALL". Evaluates to FALSE if any member of the set is NULL.	SELECT * FROM employees WHERE salary NOT IN (SELECT salary FROM employees WHERE department_id = 30); SELECT * FROM employees WHERE job_id NOT IN ('PU_CLERK', 'SH_CLERK');

Range condition



NULL condition

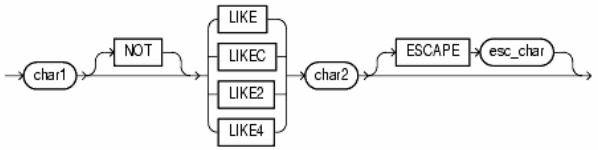


Condition	Operation	Example
IS [NOT] NULL	Tests for nulls. This is the only condition that you should use to test for nulls.	SELECT last_name FROM employees WHERE commission_pct IS NULL;

EXISTS condition

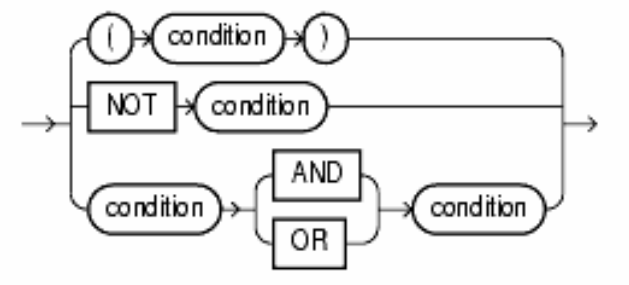
Condition	Operation	Example
EXISTS	TRUE if a subquery returns at least one row.	SELECT department_id FROM departments d WHERE EXISTS (SELECT * FROM employees e WHERE d.department_id = e.department_id);

LIKE condition



- LIKE : chaîne de caractères
  - LIKEC : codage UNICODE
  - LIKE2 : codage USC2 et LIKE4 : codage USC4
- Joker
  - _ n 'importe quel caractère (unique)
  - % n 'importe quelle chaîne de caractères

Compound condition



Conclusion

- Importance des expressions
  - un peu partout
- Importance des conditions
  - parfois difficultés de rédaction
  - importance car SQL langage déclaratif